Chapter2

Neighborhood-Based Collaborative Filtering

2.1 Introduction

- memory-based algorithms이라고도 불리는 Neighborhood-Based cf는 초기의 cf중 하나이다. 이 알고리즘은 rating behavior의 비슷한 패턴과 비슷한 items에 비슷한 ratings을 부여한 비슷한 user에 근거한다.
- neighborhood-based algorithms의 두 주요 유형

  → user를 neighboring 하는지, item을 neighboring하는지에 따라 나뉨

  1. User-based collaborative filtering : user A와 비슷한 user들이 A를 위한 추천에 사용된다. 각 item의 rating은 "peer group" ratings의 weighted average로 계산된다
  2. Item-based collaborative filtering : item B에 대한 추천을 하기 위해서 B와 비슷한 item set *S*를 만든다. 그후, user A의 item B에 대한 rating을 예측하기 위해 S에서 A에 의해 rating된 것을 토대로 weighted average를 한다.

- 예측의 두가지 형태
  1. Predicting the rating value of a user-item combination : missing rating을 예측한다.
  2. Determining the top-k items or top-k users : top-k items를 찾는 것이 top-k user를 찾는 것에 비해 더 흔히 사용되다. top-k items을 찾는 것은 실제 추천 과정과 비슷해서 도움이 되고, top-k users를 찾는 것은 마케팅을 하는 데에 도움이 된다.
     - 이 과정을 수행하기 위해선 대부분 1인 rating 예측이 필요하다. 때문에 rating을 미리 계산해 놓음으로써 효율성을 높이기도 한다.

2.2 Key Properties of Rating Matrices

- Rating을 정의하는 다양한 방법

1. Continuous ratings : rating이 연속형으로 정의되어 있는 형태이다. user에게 실수를 생각해야 하는 부담감을 주기 때문에 잘 사용되지 않는다 ex)-10~10
2. Interval-based ratings : 5점척도/10점척도 등의 rating 형태. 이 때, interval은 각 rating의 정수마다 동일해야 한다. ex)-2 to 2 / 1 to 5
3. Ordinal ratings : 값이 categorical(e.t., 매우 좋다, 좋다) 는 점을 제외하면 interval-based rating과 같다. interval-based rating과 달리 interval이 동일하지 않지만, 대부분 매우좋다→5점 과같이 변경할 수 있기 때문에 거의 차이가 없다. 대부분 긍부정의 수를 같게 하여 편향을 없앤다. 짝수개수의 척도를 사용하여 중립적인 의견을 없앤 것을 *forced choice* method라고 한다.
4. Binary ratings : 긍/부정의 두가지 의견밖에 없는 형태
5. Unary ratings : 긍정적인 선호만들 허락하는 rating.

- real-world의 rating frequency distribution
  - 실제로는 인기있는 item의 경우에만 rating이 많고 거의 대부분의 item의 rating은 수가 굉장히 적다. 따라서 long-tail인 highly skewed distribution을 갖는다.
  - 이러한 분포를 통해 알 수 있는 중요한 결과
    1. 실제로 인기있는 item에 비해 인기가 적은(덜 경쟁적인) item이 더 큰 이익을 가져오기 때문에 lower frequency items을 추천함으로써 이익을 실현한다.
    2. long tail에서의 적은 observed rating으로 long tail에서 robust rating prediction이 어렵다. 실제로 많은 추천은 인기있는 item을 추천하는 경향이 있다.
    3. infrequent item보다 인기있는 item을 통해 neighborhood가 정해지는 경향이 있다. 이것은 잘못된 결과를 이끌기도 한다.

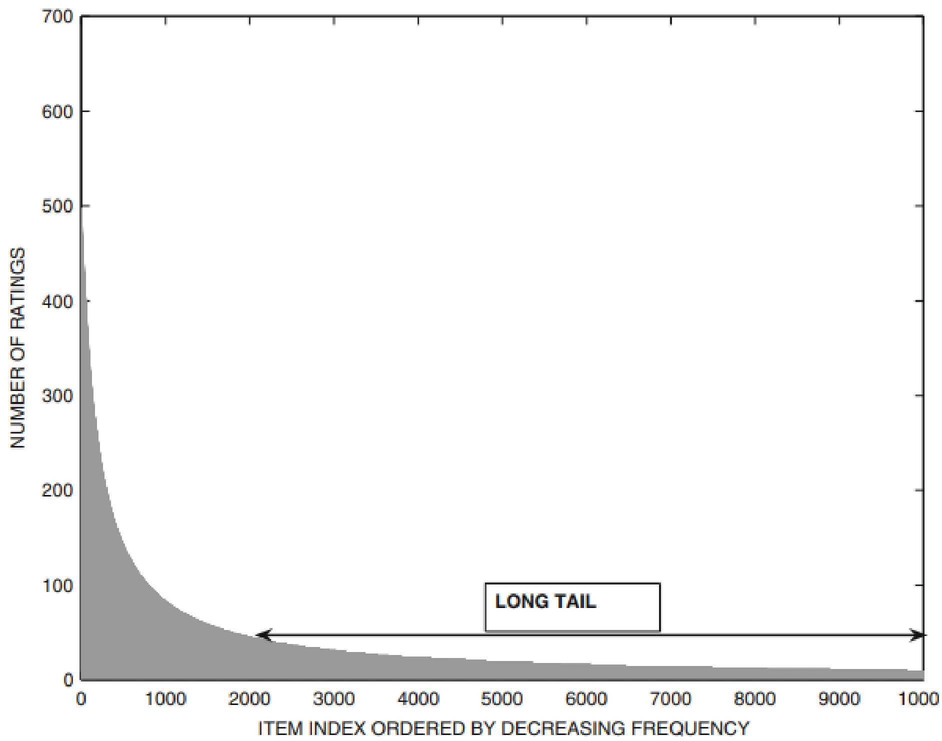


Figure 2.1: The long tail of rating frequencies

2.3 Predicting Rating with Neighborhood-Based Methods

- user-item matrix를 활용하여 similarity를 계산한다. 이때, user-based/item-based인지에 따라 row방향인지 column방향인지가 다르다.

Table 2.1: User-user similarity computation between user 3 and other users

Item-Id ⇒ / User-Id ⇓	1	2	3	4	5	6	Mean Rating	Cosine$(i, 3)$ (user-user)	Pearson$(i, 3)$ (user-user)
1	7	6	7	4	5	4	5.5	0.956	0.894
2	6	7	?	4	3	4	4.8	0.981	0.939
3	?	3	3	1	1	?	2	1.0	1.0
4	1	2	2	3	3	4	2.5	0.789	-1.0
5	1	?	1	2	3	3	2	0.645	-0.817

Table 2.2: Ratings matrix of Table 2.1 with mean-centering for adjusted cosine similarity computation among items. The adjusted cosine similarities of items 1 and 6 with other items are shown in the last two rows.

Item-Id ⇒ / User-Id ⇓	1	2	3	4	5	6
1	1.5	0.5	1.5	-1.5	-0.5	-1.5
2	1.2	2.2	?	-0.8	-1.8	-0.8
3	?	1	1	-1	-1	?
4	-1.5	-0.5	-0.5	0.5	0.5	1.5
5	-1	?	-1	0	1	1
Cosine$(1, j)$ (item-item)	1	0.735	0.912	-0.848	-0.813	-0.990
Cosine$(6, j)$ (item-item)	-0.990	-0.622	-0.912	0.829	0.730	1

2.3.1 User-Based Neighborhood Models

- similarity를 계산한 후 rating을 계산한다. similarity와 rating을 계산하는 다양한 방법이 존재한다.

1. similarity 계산

   → similarity를 계산함으로써 "peer group"을 찾는다.

   - 파라미터
     - $R = [r_{uj}]$: user-item의 mxn의 matrix
     - I_u : user u가 specified한 item 목록 ex) $I_u = [1, 3, 5]$
   - $I_u \cap I_v$ set을 이용하여 similarity $Sim(u, v)$를 계산한다.(user들은 각기 다른 item에 rating 하기 때문)
     - Pearson 상관계수

$$\mathrm{Sim}(u, v) = \mathrm{Pearson}(u, v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}}$$

- μ_u / μ_v 를 계산할 때, user u와 v 둘다 rating한 item의 rating만을 사용할 수도, 각 user의 모든 rating을 사용할 수도 있다. 엄밀하게 하면, 두 user 모두 rating한 item의 rating을 사용하여야 하지만, 그러한 item이 적다면 모든 rating을 사용하는 방법이 더 나을 수도 있다.
- user마다 예측한 item이 다르기 때문에, item마다 "peer group"이 다를 수 있다.

2. rating 계산

→ peer group의 rating을 weighted average함으로써 rating을 예측한다.

- mean centered
  - 각 user마다 rating scale이 다를 수 있기 때문에 mean-centered를 통해서 scale을 맞춰준다.(어떤 user는 대부분의 item을 좋아하고, 어떤 user는 대부분의 item을 싫어할 수도 있기 때문)
  - $s_{uj} = r_{uj} - \mu_u \; \forall \mu \in \{1...m\}$
- predict
  - mean centered rating을 이용하여 weighted average를 구한다.
  - user u의 item j에 대한 similarity group을 $P_u(j)$ 라고 했을 때, prediction function은 다음과 같다.

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \mathrm{Sim}(u,v) \cdot s_{vj}}{\sum_{v \in P_u(j)} |\mathrm{Sim}(u,v)|} = \mu_u + \frac{\sum_{v \in P_u(j)} \mathrm{Sim}(u,v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\mathrm{Sim}(u,v)|}$$

  - 예측했을 당시, rating 범위에서 벗어날 수도 있기 때문에 가까운 rating으로 보정하는 과정이 필요하다.

2.3.1.1 Similarity Function Variants

- raw ratings

  → mean-centering함으로써 bias adjustment effect 가 있기 때문에 일반적으로 피어슨 상관계수가 더 흔히 사용된다.

  - mean-centered한 adjusted cosine을 사용할 수도, raw data를 사용한 raw cosine을 사용할 수도 있다.

- 두 user 모두 rating한 item으로 normalization할 수도, 각 user에 대해서 할 수도 있다.

$$\text{RawCosine}(u, v) = \frac{\sum_{k \in I_u \cap I_v} r_{uk} \cdot r_{vk}}{\sqrt{\sum_{k \in I_u \cap I_v} r_{uk}^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} r_{vk}^2}}$$

$$\text{RawCosine}(u, v) = \frac{\sum_{k \in I_u \cap I_v} r_{uk} \cdot r_{vk}}{\sqrt{\sum_{k \in I_u} r_{uk}^2} \cdot \sqrt{\sum_{k \in I_v} r_{vk}^2}}$$

- significance weighting

  → user u와 v가 공통으로 rating한 수가 너무 적은 경우, 해당 similarity의 신뢰도는 낮다. 따라서 discount factor을 도입함으로써 해당 쌍에 대한 중요성을 낮춘다. 임계점을 β로 정하고, 이 수 보다 적은 공통 rating items을 가지는 경우 discount한다.

  - the discount factor
    - $\frac{min\{|I_u I_v|, \beta\}}{\beta}$, 항상 0~1사이의 값을 가짐

$$\text{DiscountedSim}(u, v) = \text{Sim}(u, v) \cdot \frac{\min\{|I_u \cap I_v|, \beta\}}{\beta}$$

  - discounted similarity를 통해 peer group을 정하고, 이를 prediction하는 데에도 사용한다.

2.3.1.2 Variants of the Prediction Function

- mean-centering 대신 Z-score z_{uj}를 사용할 수도 있다. final prediction 후에 역계산을 통해 rating을 산출해야 한다. rating의 범위에서 벗어난 예측을 자주 하는 단점이 있고, mean-centering과 Z-score 중 어떤 것이 성능이 더 좋은지는 연구 중에 있다.

$$\sigma_u = \sqrt{\frac{\sum_{j \in I_u} (r_{uj} - \mu_u)^2}{|I_u| - 1}} \quad \forall u \in \{1 \dots m\}$$

$$z_{uj} = \frac{r_{uj} - \mu_u}{\sigma_u} = \frac{s_{uj}}{\sigma_u}$$

$$\hat{r}_{uj} = \mu_u + \sigma_u \frac{\sum_{v \in P_u(j)} \text{Sim}(u,v) \cdot z_{vj}}{\sum_{v \in P_u(j)} |\text{Sim}(u,v)|}$$

- similarity를 amplify
  - $Sim(u,v) = Pearson(u,v)^{\alpha}$
  - α로 상관계수의 값을 증폭시킨 값을 similarity로 이용. 만약 $\alpha > 1$ 이라면 weighting시에 similairty의 중요성을 증폭시킨 것에 해당
- rating voting
  - rating average 하는 대신 rating value를 categorical하게 나누고 voing하는 방식을 사용한다. peer group은 각 rating이 voting함으로써 rating을 예측한다.
  - 더 likely rating을 제공할 수 있으며, 부 rating을 제공할 수 있으며, distinct rating의 개수가 적을 때 더 효과적이다. 또한, ordinal rating에서 효과적이다. 하지만 distinct rating의 개수가 많다면 덜 robust하고 rating사이의 순서에 대한 정보를 잃을 것이다.

2.3.1.3 Variations in Filtering Peer Groups

- top-k users를 peer groups으로 선정하게 되면, 약하거나 음의 상관관계를 갖는 user들이 peer groups으로 선정되어 예측력을 저하시킬 수 있다. 따라서 이러한 user를 filter out해주는 작업을 해주기도 한다.

2.3.1.4 Impact of the Long Tail

- Long tail 분포를 가지면서, peer group을 계산하고 rating을 계산할 때 user간에 구별이 적어지는 문제가 생길 수 있다. 따라서 Inverse Document Frequency(idf)와 비슷한 Inverse User Frequency(iuf)를 사용함으로써 이러한 문제를 해결한다.
- 각 item의 가중치와 수정된 피어슨 상관계수
  - m_j: item j의 rating 개수
  - m : user의 전체 수

$$w_j = \log\left(\frac{m}{m_j}\right) \quad \forall j \in \{1 \dots n\}$$

$$\text{Pearson}(u,v) = \frac{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{vk} - \mu_v)^2}}$$

2.3.2 Item-Based Neighborhood Models

- User-Based models과 동일한 과정을 거쳐 mean-centered rating인 s_{uj}를 얻고, item i를 rating한 user set인 U_i를 얻는다. 그 후, user u가 rating하고 item t와 비슷한 top-k item $Q_t(u)$를 통해 weighted average를 계산한다.
- User-based models의 다양한 variations은 item-based models에서 또한 사용가능하다.
- 자기 자신의 rating을 사용하여 rating을 예측하기 때문에 보다 consistent한 예측이 가능하다. 예를들어, rating의 범위에서 잘 벗어나지 않는다.

$$\text{AdjustedCosine}(i,j) = \frac{\sum_{u \in U_i \cap U_j} s_{ui} \cdot s_{uj}}{\sqrt{\sum_{u \in U_i \cap U_j} s_{ui}^2} \cdot \sqrt{\sum_{u \in U_i \cap U_j} s_{uj}^2}}$$

2.3.3 Efficient Implementation and Computational Complexity

- prediction하고 ranking하는 중간 과정에서 수치가 재사용되기 때문에 offline에 중간 계산 값을 저장하고 다음 ranking process에서 사용하는 것이 좋다. offline phase에서 시간을 소요하여 peer group을 정해놓으면 online prediction에서 효율적으로 이를 사용할 수 있다.
- offline phase : similarity를 계산하고 user/item의 peer group을 사전 저장한다.
  - 각 user의 specified rating이 $n' \ll n$, 각 item의 specified rating이 $m' \ll m$ 이라면,
    - 한 user의 peer group을 구하는 시간복잡도 : $O(m \cdot n')$
    - 모든 user의 peer group을 구하는 시간복잡도 : $O(m^2 \cdot n')$
    - 모든 item의 peer group을 구하는 시간복잡도 : $O(n^2 \cdot m')$
    - 각 user의 similarity를 저장하는데 필요한 공간 : $O(m^2)$
    - 각 item의 similarity를 저장하는데 필요한 공간 : $O(n^2)$

  (보통 user의 수가 item의 수보다 많아서 저장하는데 필요한 공간이 더 많음

- online phase : 저장된 peer group을 통해 prediction한다. (k : neighborhood size)
    - top-r item을 정할 때의 시간복잡도 : $O(k\cdot n)$
    - top-r user를 정할 때의 시간복잡도 : $O(k\cdot m)$

2.3.4 Comparing User-Based and Item-Based Methods

- Item-Based Methods는 user's *own* ratings을 사용하여 rating prediction을 하기 때문에 종종 더 정확한 예측을 하고, *shilling attacks*에 더 robust하다. 또한 추천의 이유를 제공할 수 있다. (e.g., Because you watched "A", [the recommendations are] <List>) 마지막으로, 잘 변하지 않는 안정적인 예측이 가능하다. 이는 첫째로, 일반적으로 user의 수가 item의 수보다 많아서, 어떤 user가 rating한 item의 수 보다는 어떤 item에 rating한 user의 수가 더 많기 때문이다. 둘째로, 새로운 user가 새로운 item보다 더 자주 유입되기 때문에, 새로운 user로 item의 peer goup은 잘 바뀌지 않지만, user peer group은 자주 계산될 필요가 있기 때문이다.
- User-Based Methods는 Item-Based Methods에 비해 추천되는 item의 다양성이 존재한다. 이를 통해 user에게 재미와 흥미를 유발할 수 있다. privacy 문제 때문에 추천의 이유를 제공하는 데에 제약이 있다.

2.3.5 Strengths and Weaknesses of Neighborhood-Based Methods

- 간단하고 직관적이기 때문에 예측을 수행과 디버깅, 그리고 결과를 해석하는데에 좋다. 또한 상대적으로 새로운 user/item에 대해 stable하다.
- offline-phase에서 시간과 공간(hardware)이 많이 소요된다.
- matrix가 sparse하기 때문에, coverage에 한계가 있다. (예를들어 user A에 대한 item B rating prediction을 하고싶어도 user A의 neighborhood 중 item B를 rating한 사람이 아무도 없다면 rating prediction을 할 수 없다. 하지만 top-k prediction이라면, item B를 추천하지 않으면 된다.)또한 sparse하기 때문에 두 user 간의 공통 rating item이 몇 개 없는 상황이 발생하고, 이는 similarity가 robust하지 않은 문제를 유발한다.

2.3.6 A Unified View of User-Based and Item-Based Methods

- User-Based Methods와 Item-Based Methods를 통합해서, 즉 row와 column을 결합하여 이용함으로써 성능을 향상시킨다. 이 때 다양한 combination function을 사용할 수 있다. 한 논문에서는 context-sensitive recommender systems의

multidimensional model을 사용하고, 이는 user, item 그리고 다른 contextual 차원들이 통합되어 있는 단일 framework다.

1. target entry (u, j)의 similar entries를 결정하기 위하여 row와 column간의 similarity를 combination function을 사용하여 계산한다. 예를들어 행간의, 열간의 cosine similarity를 계산한 후 이를 합침으로써 similar entries를 결정할 수 있다.
2. 1에서 계산한 similar entries의 weighted combination을 사용하여 target entry를 예측한다.

2.4 Clustering and Neighborhood-Based Methods

- offline-phase에서의 시간복잡도를 줄이기 위해 clustering 을 이용하여 peer group을 구한다. 각 point들의 peer group은 자신이 속해있는 cluster이다.(그 point가 cluster의 center point가 아닐지라도)
  - 효율적이지만 정확도에서의 일부 손실이 생길 수 있다.
  - user-based, item-based 모두에서 사용 가능하다.
- K-means clustering
  - Euclidean distance or the Manhattan distance등을 사용. 보통 Manhattan distance 사용
    - Manhattan distance는 그리드 형태의 배열에서의 거리, 차원의 수가 굉장히 많아 차원의 저주가 발생할 가능성이 높을 때 사용하면 유용
      - 차원의 저주 : 차원이 증가할수록 공간의 부피가 기하급수적으로 증가하기 때문에 동일한 개수의 데이터 밀도는 차원이 증가할 수록 급속도록 sparse해지는 현상
        - 유클리디언 거리는 가깝지만 실제로는 먼 경우가 발생할 수 있음. ex) 매니폴드 현상
  - 불완전한 행렬이기 때문에 각 point와 centroids엔 특정되지 않은 feature가 존재

    → 각 cluster centroids와 동시에 존재하는 feature만으로 distance를 적용

    → 산출된 distance를 동시에 존재하는 feature개수로 나눔
- co-clustering을 이용하여 row, column을 동시에 사용할 수도 있다.

2.5 Dimensionality Reduction and Neighborhood Methods

- Dimensionality reduction은 latent factor models이라고도 불린다. latent vector를 사용하여 distance를 구하고 peer group을 구할 수 있다.
    - 효율적이고 정확성을 높일 수 있다.
- 두 가지 방법
    1. row-wise *or* column-wise latent factor를 만든다. 이런 방법으로 item 차원 혹은 user 차원을 줄인 latent representations를 생성하고, neighborhood algorithms에 사용한다.
    2. row space와 column space 둘다의 latent representations을 생성한다. 이를 통해 neighborhood-based method 없이, 전체 rating matrix를 예측한다.
- 방법1(row/column 방향)
    - 저차원의 full-specified vector를 생성할 수 있다.

      → robust, 효율성 상승

      → user-based와 item-based 모두 사용 가능
    - PCA/SVD를 사용할 수 있다. 예를들어, R = mxn의 행렬이 있으면,
        - SVD
            - $S = R_f^T R_f$: similarity(nxn) matrix 생성 후, 이를 이용
            - $S = P \triangle P^T$에서 P의 차원을 d로 줄임으로써 차원을 축소
                - $\triangle$: diagonal matrix(대각행렬)
                - P : orthonormal eigenvectors(정규직교행렬)
        - PCA
            - R의 공분산행렬을 구하고, 이를 이용하여 차원을 축소한다.

      → SVD와 PCA는 column 방향으로 mean-centered되어 있으면 같은 방식에 해당.

      → 대안적으로 row를 mean center하고 난 후, 각 column을 mean center할 수도 있다.

2.5.1 Handling Problems with Bias

	Godfather	*Gladiator*	*Nero*
Godfather	2.55	4.36	2.18
Gladiator	4.36	9.82	3.27
Nero	2.18	3.27	3.27

Table 2.3: Example of bias in estimating covariances

User Index	*Godfather*	*Gladiator*	*Nero*
1	1	1	1
2	7	7	7
3	3	1	1
4	5	7	7
5	3	1	?
6	5	7	?
7	3	1	?
8	5	7	?
9	3	1	?
10	5	7	?
11	3	1	?
12	5	7	?

- mean-filling technique
    - missing value를 단순히 평균으로 채우게 되면, bias가 생겨 data를 잘 반영하지 못한다. 따라서 새로운 방법이 필요하다.
        - 예를들어 Gladiator과 Nero의 rating 4개가 모두 일치함에도, missing value를 평균으로 채우게 되면 covariance이 낮게 나온다.

2.5.1.1 Maximum Likelihood Estimation(MLE)

- EM-algorithm사용
- Simplified Model - averaged normalization
    - 공통된 specified rating만을 가지고 covariance를 계산한다. 만약 공통된 rating user가 없다면 두 item의 covariance는 0이다. 이를통해 bias를 줄일 수 있다.
    - 계산된 covariance matrix를 통해 $A = [a_{ui}]_{m \times d}$를 구한다.

      → user마다 rating한 개수가 다를 때 특히 더 유용하다.

      → missing value를 채울 수 있고, 차원축소가 가능하다.

      → user방향으로 하면 user의 수를 매우 줄일 수 있기 때문에 item-based에서 특히 더 유용하다.

$$a_{ui} = \frac{\sum_{j \in I_u} r_{uj} e_{ji}}{|I_u|}$$

- e_i : i번째 eigenvector
- a_{ui}: PCA를 통해 새로 구해진 행렬의 u행 i열 원소

- r_{uj} : user u의 item j의 rating
- user u의 i번째 vector에 대한 contribution
- $|I_u|$: user u가 rating한 item의 개수

→ but, covariance matrix estimation이 robust한 estimation을 하기 위해서는 상당한 observed rating이 필요하기 때문에 매우 sparse할 경우 적합하지 않음

→ 직접적으로 utility를 구할 수 있다.

2.5.1.2 Direct Matrix Factorization of Incomplete Data

- SVD를 이용하여 mxn 크기의 fully specified rating matrix를 구한다.

  $R = Q\Sigma P^T$

  - Σ : singular values
- truncated SVD
  - d ≤ min{m,n}인 d개의 가장 큰 singular values만을 사용
  - $R \sim Q_d \Sigma_d P_d^T$
- missing value를 observed entries를 이용하여 mse를 최소화함으로써 문제를 풀 수 있다. 이 때, nonlinear optimization techniques로 푸는 것도 가능하다.

→ robust 하고 unbiased lower dimensional representation을 얻을 수 있다.

→ 직접적으로 utility를 구할 수 있다.

2.6 A Regression Modeling View of Neighborhood Methods

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \text{Sim}(u,v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\text{Sim}(u,v)|}$$

$$\hat{r}_{ut} = \frac{\sum_{j \in Q_t(u)} \text{AdjustedCosine}(j,t) \cdot r_{uj}}{\sum_{j \in Q_t(u)} |\text{AdjustedCosine}(j,t)|}$$

- neighborhood methods를 모든 item, 모든 user의 rating를 이용하는 것으로 넓혀서 보면 linear regression model과 같다. similarity는 heuristic하게 구한 것만 다를

뿐, linear regression의 combination weights의 역할을 한다. peer-group의 것들은 heuristic weights를 갖고, 그 밖의 것들은 0의 weights를 갖는다.

- 즉, *neighborhood-based models 은 linear regression models의 heuristic variants 이다*. item-based는 column의 linear combination이고, user-based는 row의 linear combination이다.
- similarity를 heuristic 하고 arbitrary하지 않고 combination weights로 보게 되면, 변수간의 상호의존성을 포함할 수 있게 된다. 예를들어 어떤 item set을 비슷한 방식으로 rating했다면 이 item들의 계수 또한 상호의존적이다.
- analogous regression-based model, 등 다양한 모델을 사용할 수 있다.

→ 수학적인 최적화 모델에서 기초하기 때문에 combination weights는 더 잘 판단될 수 있다.

2.6.1 User-Based Nearest Neighbor Regression

- $P_u(j)$ 를 정의가 기존의 정의와 차이가 있다. 따라서 기존의 경우에 비해 $P_u(j)$ 가 k 명보다 상당히 작다. $P_u(j)$ 는 Pearson coefficient로 구한다.
  - 기존 : item j에 rating한 가장 비슷한 k명의 user
  - 변경 후 : 가장 비슷한 k명 중 item j에 rating한 user
- $\hat{r}_{u,j} = \mu_u + \sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - \mu_v)$
  - w_{vu}^{user} : parameter
    - w_{uv}^{user} 와는 다를 수 있다. $P_u(j)$ 에 있는 user 외에는 값이 0이기 때문에 추정할 계수 값이 줄어든다.
- 목적함수 - 각 user에 따라 다름
  - decomposed formulation(한 user u에대한 목적함수)

$$\text{Minimize } J_u = \sum_{j \in I_u} (r_{uj} - \hat{r}_{uj})^2$$
$$= \sum_{j \in I_u} \left(r_{uj} - \left[\mu_u + \sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - \mu_v) \right] \right)^2$$

  → 독립적으로 사용된다면 decomposed formulation을 사용

  - consolidated formulation(모든 user에 대한 목적함수)

$$\text{Minimize} \sum_{u=1}^{m} J_u = \sum_{u=1}^{m} \sum_{j \in I_u} \left(r_{uj} - \left[\mu_u + \sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - \mu_v) \right] \right)^2$$

→ 분해가 가능하지 않은 optimization models(e.g., matrix factorization)을 사용한다면 이러한 formulation을 사용

2.6.1.1 Sparsity and Bias Issues

- sparse하여 같은 user의 item마다 peer group의 size가 크게 다르기 때문에, user u가 rating한 item j에 rating한 peer group size에 영향을 많이 받는다. 만약 어떤 user u가 rating한 item에 rating한 peer group size가 1이라면, 그 한명의 coefficient는 peer group size가 1이라는 사실에 많은 영향을 받을 것이다.
- 이러한 문제를 해결하기 위해, regression coefficients는 target user의 모든 peer에 기초한다고 가정한다. $\frac{P_u(j)}{k}$로 수정한다. 또한, 불완전한 정보를 보강해야 한다.

  <방법1>

$$\hat{r}_{uj} \cdot \frac{|P_u(j)|}{k} = \mu_u + \sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - \mu_v)$$

  <방법2>

$$\hat{r}_{uj} = b_u^{user} + \frac{\sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - b_v^{user})}{\sqrt{|P_u(j)|}}$$

  - k는 constant factor이기 때문에 k 생략.(optimization과정에서 알아서 반영됨
  - constant offset μ_v를 bias variable b_u로 대체
  - non-linear model

  <방법3>

$$\hat{r}_{uj} = b_u^{user} + b_j^{item} + \frac{\sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - b_v^{user} - b_j^{item})}{\sqrt{|P_u(j)|}}$$

  - item bias를 추가

2.6.2 Item-Based Nearest Neighbor Regression

$$\hat{r}_{ut} = \sum_{j \in Q_t(u)} w_{jt}^{item} \cdot r_{uj}$$

- $Q_t(u)$ 를 정의가 기존의 정의와 차이가 있다. 따라서 기존의 경우에 비해 $Q_t(u)$ 가 k명보다 상당히 작다. $Q_t(u)$ 는 adjusted cosine으로 구한다.
    - 기존 : item t와 비슷한 user u가 rating한 k개의 item
    - 변경 후 : item t와 비슷한 k개의 item 중 user u가 rating 한 item
- 목적함수
    - decomposed formulation(한 item에 대한 목적함수)

$$\begin{aligned}\text{Minimize } J_t &= \sum_{u \in U_t} (r_{ut} - \hat{r}_{ut})^2 \\ &= \sum_{u \in U_t} (r_{ut} - \sum_{j \in Q_t(u)} w_{jt}^{item} \cdot r_{uj})^2\end{aligned}$$

    - consolidated formulation(모든 item에 대한 목적함수)

$$\text{Minimize } \sum_{t=1}^{n} \sum_{u \in U_t} (r_{ut} - \sum_{j \in Q_t(u)} w_{jt}^{item} \cdot r_{uj})^2$$

    - J_t에 regularization term을 추가하기도 함(overfitting 방지)

$$\lambda \sum_{u \in U_t} \sum_{j \in Q_t(u)} (w_{jt}^{item})^2$$

- sparse해서 일어나는 문제를 해결하기 위해

  <방법1>

$$\hat{r}_{ut} = b_u^{user} + b_t^{item} + \frac{\sum_{j \in Q_t(u)} w_{jt}^{item} \cdot (r_{uj} - b_u^{user} - b_j^{item})}{\sqrt{|Q_t(u)|}}$$

  <방법2>

$$\hat{r}_{ut} = b_u^{user} + b_t^{item} + \frac{\sum_{j \in Q_t(u)} w_{jt}^{item} \cdot (r_{uj} - B_{uj})}{\sqrt{|Q_t(u)|}}$$

- $b_u^{user} + b_j^{item}$ 을 B_{uj}의 constant term으로 통합
    - B_{uj}는 global mean이고, non-personalized approach를 사용하여 도출됨
    - model을 build하기 전에 구해놓아야 함

2.6.3 Combining User-Based and Item-Based Methods

- user-based와 item-based를 합쳐서 similar user와 similar item으로 추정한다. 개별적으로 model을 사용하는 것에 비해 성능이 더 좋다.

$$\hat{r}_{uj} = b_u^{user} + b_j^{item} + \frac{\sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - B_{vj})}{\sqrt{|P_u(j)|}} + \frac{\sum_{j \in Q_t(u)} w_{jt}^{item} \cdot (r_{uj} - B_{uj})}{\sqrt{|Q_t(u)|}}$$

- 목적함수
    - SSE를 최소화하는 것을 목표로 한다.
    - decomposed formulation은 사용하지 않는다.

2.6.4 Joint Interpolation with Similarity Weighting

- user-based model에 의해 예측된 target user u의 rating을 *똑같은* item의 rating과 비교하는 것이 아닌, *다른* item의 rating과 비교한다.
- 목적함수

$$S = \{(u, t) : r_{ut} \text{ is observed}\}$$

$$\begin{aligned}
\text{Minimize } & \sum_{s:(u,s) \in S} \sum_{j:j \neq s} AdjustedCosine(j, s) \cdot (r_{us} - \hat{r}_{uj})^2 \\
&= \sum_{s:(u,s) \in S} \sum_{j:j \neq s} AdjustedCosine(j, s) \cdot \left(r_{us} - \left[\mu_u + \sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - \mu_v) \right] \right)^2
\end{aligned}$$

- $P_u(j)$: 기존의 정의를 활용한다.(item j에 rating한 사람 중 target user u과 가까운 k명)
- similarity의 활용
  - item-item similarities는 i예측된 item의 비슷한 item의 rating과 비슷하도록 하는 multiplicative factor로 사용된다.
  - user-user similarities는 coefficient를 peer group에만 제한되도록 하여, rating을 예측하는데 사용된다.
  - user와 item의 역할을 바꿀 수도 있지만 원래 방법대로 하는 것이 더 성과가 좋다.

2.6.5 Sparse Linear Models(SLIM)

- sparse linear model은 regularization methods사용으로 coefficients를 sparsity하게 한다.
- 값이 비음수여야 하기 때문에, mean-centering하면 안된다.(mean-centering이 음수값을 만들기 때문). implicit feedback metrics(e.g., click-through data or sales data) 에 적용하기 가장 적절하다.(임의의 rating matrics에도 사용 가능하긴 하지만, non-negative rating matrics일 때 주요 이점이 실현됨)
- missing value를 초기에 0으로 설정하고 예측된 값으로 item을 rank한다.
- coefficients를 peer group에 제한하지 않는다. 하지만, 자기 자신과의 coefficient는 포함하지 않아야 한다. 따라서 $w_{tt}^{item} = 0$의 제약조건을 넣는다.

$$\hat{r}_{ut} = \sum_{j=1}^{n} w_{jt}^{item} \cdot r_{uj} \quad \forall u \in \{1 \dots m\}, \ \forall t \in \{1 \dots n\}$$

<matrix 기반 formulation>

$$\hat{R} = RW^{item}$$
$$\text{Diagonal}(W^{item}) = 0$$

  - $W^{item} = [w_{jt}^{item}]$, $R = [\hat{r}_{uj}]$
- 목적함수

  $||R - RW^{item}||^2$

$$\begin{aligned}\text{Minimize } J_t^s &= \sum_{u=1}^{m}(r_{ut}-\hat{r}_{ut})^2 + \lambda \cdot \sum_{j=1}^{n}(w_{jt}^{item})^2 + \lambda_1 \cdot \sum_{j=1}^{n}|w_{jt}^{item}| \\ &= \sum_{u=1}^{m}(r_{ut}-\sum_{j=1}^{n} w_{jt}^{item} \cdot r_{uj})^2 + \lambda \cdot \sum_{j=1}^{n}(w_{jt}^{item})^2 + \lambda_1 \cdot \sum_{j=1}^{n}|w_{jt}^{item}| \\ &\text{subject to:} \\ &w_{jt}^{item} \geq 0 \quad \forall j \in \{1 \dots n\} \\ &w_{tt}^{item} = 0\end{aligned}$$

- 비음수 값을 만들기 위해 coefficient 또한 비음수로 제약
  - 각 coefficient를 영향을 미치는 정도로 해석 가능
- L1, L2 regularization term 추가.
  - L1 regularization으로 sparse solution을 내도록 한다.
- coordinate descent method를 사용한다.

- 이전의 Linear Regression model과의 차이점
  1. k개의 coefficient에 제한되지 않는다. 만약 모든 user와 관련있다면 모든 user의 coefficient는 양수일 수 있다. 즉, 이전의 linear regression model은 heuristic approach로 feature selection을 하는 반면, SLIM은 model을 통해 learning하는 방식으로 feature selection을 한다.
  2. implicit feedback dataset을 위해 디자인된 모델이기 때문에 비음수 값만을 가진다. 따라서 오직 긍정적이 선호 지표(e.g., 구매한 상품 개수) 만 있는 case에도 사용할 수 있다. 하지만 arbitrary rating에는 적절하지 않다. rating이 호불호를 나타내면 비음수의 부분합을 해석성을 잃는다. 예를들어 두개의 싫어하는 rating은 좋아하는 rating을 구성한다고 해석할 수 없다.
  3. SLIM의 coefficient는 비음수이다. implicit dataset을 위한 model이기 때문이다. 이를 통해 coefficient는 더 해석가능하고, 정확성을 높이기도 한다. 하지만 몇 논문에서는 이러한 제약을 없애는 것이 더 좋은 성능을 가져온다고 주장한다.
  4. SLIM은 predicted value의 순서에 따라 ranking함으로써 사용한다. predicted value는 초기 0으로 설정한 것의 error라고 볼 수 있기 때문에, 그 값을 ranking할 수 있다.
  5. specified rating을 heuristic adjustment factors로 조정해서 사용하지 않는다. 예를들어 $\sqrt{|Q_t(u)|}$ factor를 사용하지 않는다. missing value를 0으로 채워도 rating이 호불호를 나타내는 case에 비해 bias가 훨씬 적기 때문이다.

2.7 Graph Models for Neighborhood-Based Methods

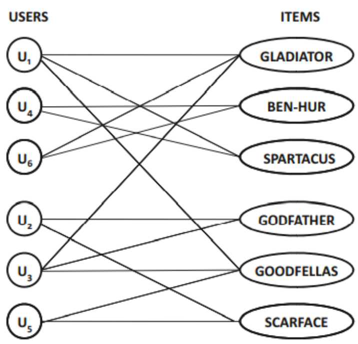
- 많은 graph models은 structural transitivity/ ranking techniques를 사용하여 sparse함을 극복하고 similaity를 정의한다. 많은 알고리즘을 이용하여 user 간 / item 간 / 둘다 의 관계를 구조적으로 표현할 수 있다.

2.7.1 User-Item Graphs

- user와 item간의 관계를 구조적으로 표현한다. sparse rating matrices에서 더 효과적이다.
- $G = (N_u \cup N_i, A)$
    - N_u, N_i = user의 node, item의 node
    - A : edge(만약 user i가 item j에 rating 했다면)
    - edge의 개수는 observed entries와 같다.

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			5		2
U_2		5			4	
U_3	5	3		1		
U_4			3			4
U_5				3	5	
U_6	5		4			

(a) Ratings matrix



(b) User-item graphs of specified ratings

Figure 2.3: A ratings matrix and corresponding user-item graph

- 두 user사이에 짧은 경로가 많이 존재하는 한 두 user가 이웃으로 간주하기 위해 동일한 item에 많은 등급을 매길 필요가 없다. 따라서, 이 정의는 노드 간의 *간접* 연결 개념을 가진 이웃의 구성을 허용한다. 물론 동일한 item에 대한 많은 등급이 있더라도 이웃으로 된다.
- random-walk measures / Katz measure을 사용함으로써 indirect connectivity를 구한다.
    - random-walk measure & Katz measure은 link prediction 문제와 관련되어 있다.

2.7.1.1 Defining Neighborhoods with Random Walks

- 해당 user/item에서 시작하여 random walk(probabilistic measure)로 가장 자주 방문된 user/item의 집합을 정의한다.
- direct connection과 indirect connection 둘 다 사용할 수 있기 때문에 spare matrices에서 더 효과적이다.

2.7.1.2 Defining Neighborhoods with the Katz Measure

- Katz measure(weighted number of walks between a pair) 을 이용한다. 만약 이를 통해 neighborhood로 나왔으면 연결되어 있는 경향이 있는 것이다.

Definition 2.7.1 (Katz Measure)

- node i와 j사이의 Katz measure(0~1)

$$Katz(i,j) = \sum_{t=1}^{\infty} \beta^t \cdot n_{ij}^{(t)}$$

  - $n_{ij}^{(t)}$: node i와 j 사이의 length t의 walks의 수
  - $\beta < 1$: user-defined parameter
    - 더 긴 길이의 walks를 덜 강조하는 discount factor
    - β를 작은 값으로 설정함으로써 infinite summation이 수렴한다.

  <matrix 표현>

$$K = \sum_{i=1}^{\infty} (\beta A)^i = (I - \beta A)^{-1} - I$$

  - A가 대칭인접행렬(그래프의 연결 관계를 이차원의 행렬로 나타낸 것) 일 때의 matrix 표현
  - 수렴하기 위해선 β는 A의 가장 큰 고유값의 역수보다 커야한다.
  - K는 mxm의 행렬을 가진다(user-based 일때).
- graphs에서 diffusion kernel과 밀접하게 평가된다.
- A의 weighted version은 A를 graph의 weight matrix로 대체함으로써 계산될 수 있다. 이렇게 함으로써 user-item graph에서의 edge weight하고 싶은 경우 사용할 수 있다.

- target node와의 Katz measure가 가장 큰 top-k nodes가 그것의 이웃이 된다. 이를 통해 가중합으로 rating을 예측할 수 있다.

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \mathrm{Sim}(u, v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\mathrm{Sim}(u, v)|}$$

- Katz measure에서의 여러 variations
  1. Katz measure을 구할 때 원래 β를 통해 제한되긴 하지만, 추가로 path length에서의 maximum threshold를 설정한다.
  2. user간, item간 뿐만 아니라, user-item간의 친밀도를 측정함으로써 neighborhood methods외에 방법에서 weights로 사용가능하다.

2.7.2 User-User Graphs

- user-item graph를 이용하면 user-user connectivity가 짝수 개수이다. 따라서 user-item graphs 대신 user-user graphs를 2-hop connectivity에 근간해서 만들 수 있다. 이를 user-user predictability 라고 한다. 이러한 graph는 offline phase에 만들어 놓는다.
- user transitivity를 사용하기 때문에 sparse matrices에서도 잘 작동한다. 또한 예측 coverage가 늘어난다. 기존에는 만약 target user의 peer group에 terminator에 rating한 neighbors가 없다면 예측할 수 없는데, 이 구조를 사용하면 indirect neighbors을 사용할 수 있기 때문이다.
- 두 user사이의 edge는 더 많은 정보를 담고 있다. 2-hop connectivity는 horting과 predictability의 개념을 사용한다.
  - horting - 두 user의 specified rating의 수를 나타내기 위해 사용한다.
  - predictability - 공통된 rating의 similaity의 level을 나타내기 위하여 사용한다.

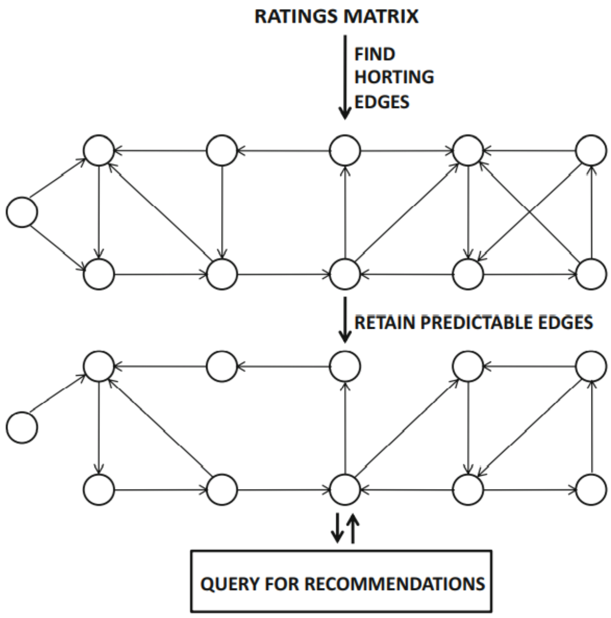


Figure 2.4: The user-user predictability approach

1. user u가 user v를 horting한다면 edge를 생성
2. user u가 user v를 horting하고 v가 u를 예측한다면 edge가 존재. 따라서 몇 edge가 dropping 됨.

Definition 2.7.2 (Horting)

- 다음 중 하나가 참일 경우 user u는 user v를 level (F,G)에서 hort한다.

$$|I_u \cap I_v| \geq F$$
$$|I_u \cap I_v| / |I_u| \geq G$$

  - F, G : 알고리즘 파라미터
  - I_u, I_v = user u가 rating한 item 개수, user v가 rating한 item 개수

Definition 2.7.3 (Predictability)

- user v의 rating을 이용하여 linear transformation function $f(\cdot)$로 user u를 예측한다. 즉 r_{uk}와 $f(r_{vk})$의 Manhattan distance에서 normalized distance로 similarity를 구한다(Manhattan segmental distance).

$$\frac{\sum_{k \in I_u \cap I_v} |r_{uk} - f(r_{vk})|}{|I_u \cap I_v|} \leq U$$

- U : 알고리즘 파라미터

- horting과 predictability의 방향은 서로 반대이다. 즉, user v가 user u를 예측하기 위해선 user u는 v를 hort 해야한다. 만약 v가 u를 예측한다면 user u에서 v로 edge가 존재한다. 이 edge에는 상응하는 linear transformation이 존재하고, 이는 *edge의 head를 예측하는데에 edge tail의 rating을 사용한다.*

- 또한 linear transformations을 전이적으로 사용함으로써 rating을 예측할 수 있다고 예측한다. 즉 user v를 통해 user u의 item k rating을 예측하고자 한다면, user u에서 user v까지의 path의 function을 순차적으로 적용한다. 이 때 가장 짧은 path로 계산한다. 이는 breadth-first algorithm이 활용된다.

$$\hat{r}_{uk}^{(v)} = (f_1 \circ f_2 \ldots \circ f_r)(r_{vk})$$

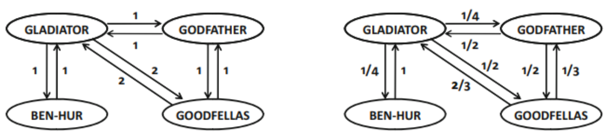
- target user u의 distance의 임계점을 D로 하였을 때의 모든 user v에 대하여 $\hat{r}_{uk}^{(v)}$을 구한 다음 평균낸다. 이를 통해 너무 긴 path 때문에 distortion하는 것을 방지한다.

2.7.3 Item-Item Graphs

- correlation graph로 item-item graphs를 그리고 이를 추천에 활용할 수 있다. 그래프를 만드는데에 rating의 values가 아닌, rating한 user의 수가 활용된다.(cosine function을 사용하여 rating value를 사용하도록 그래프를 만들 수도/ItemRank를 사용할 수도 있다.)

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS
U_1	1			5
U_2		5		
U_3	5	3		1
U_4			3	
U_5				3
U_6	5		4	

(a) Ratings matrix



(b) Unnormalized correlation graph (c) Normalized correlation graph

Figure 2.5: A ratings matrix and its correlation graphs

- item 개수 N만큼의 node가 만들어지고, 그 관계를 정의하는 edge가 존재한다. 만약 item i와 j를 동시에 rating한 user가 한명이라도 있다면 (i,j)와 (j,i) edge 모두 존재한다. 하지만 비대칭행렬이다.
- (i,j)의 weight를 계산하는 방법(U_i : item i에 rating한 user)
  1. (i,j)의 weight를 $|U_i \cap U_j|$를 초기화한다. 이 경우 weight 행렬은 대칭이다.
  2. weights를 normalized한다. w_{ij}를 node i에서 나가는 weights의 총합으로 나눔으로써 각 node에서 나가는 edge의 weights의 합을 1로 만든다. 이 때문에, 비대칭 행렬이 된다.
  3. 만들어진 그래프에 random-walk-probabilites emthods를 사용한다.
  4. neighborhood를 결정하고 이를 item-based cf에 활용한다.

2.8 Summary

- cf는 classification, regression 문제의 일반화이다. 그 중 Neighborhood-based methods는 neighborhood로 부터 rating을 예측한다.
  - user-based model : user의 neighborhood를 결정하고, 그로부터 rating을 예측.
  - item-based model : item의 neighborhood를 결정하고, target user의 rating을 이용하여 rating을 예측. 더 관련있는 예측이 가능하지만, 추천되는 item의 다양성이 줄어든다.
- heuristic한 방법으로 weight를 결정하지 않고, model 기반으로 weight를 결정함으로서, 최적화 모델을 사용할 수 있다.
- neighborhood를 결정하는데에서 발생하는 문제를 극복하기 위한 방법
  - 시간 : clustering
  - data sparsity : 차원축소, 그래프 기반 모델
    - 차원축소 :SVD, PCA 등 → 효율성과 효과성을 높일 수 있다.
    - 그래프 기반 모델 : user-item graphs, user-user graphs, item-item graphs 등 다양한 그래프를 구축하고, random-walk/shortest path methods를 사용한다.